ReadSpeaker speechEngine

ReadSpeaker US English speechEngine API
Programmer's Guide
Software Version 3.12.3

ReadSpeaker

Revision History

Date (Ver. No)	Item	Contents
2006-12 (v3.7.0)	VT_UNLOAD_UserDict_ENG()	▪ Error codes revised
	VT_GetTTSInfo_ENG()	▪ VT_DB_BUILD_DATE added
	<vtml_phoneme>	▪ alphabet revised (cmu, sapi -> x-cmu, x-sapi) ▪ alphabet added (x-pentax)
	<vtml_pause>	▪ Restriction added (+, - not usable)
	<vtml_pitch> <vtml_speed> <vtml_volume>	▪ Example revised ▪ Restriction added (+, - not usable)
	<vtml_sayas>	▪ Syntax error revised ▪ "exam" → "example" ▪ ssml:telephone revised ▪ vtml:letter revised ▪ vxml:time revised
	<vtml_sub>	▪ pos attribute deleted
	vt_eng.h	▪ VT_DB_BUILD_DATE added
	sample_eng.c	▪ #include <windows.h> added
	Appendix D	▪ x-pentax description added ▪ cmu phonetic table revised
2007-06 (v3.7.2)	VT_PLAYTTS_ENG()	▪ Detailed description added.

	VT_PLAYTTS_ENG()
	VT_STOPTTS_ENG()
	VT_PAUSETTS_ENG()
	VT_RESTARTTTS_ENG()
		▪ Functions only for Win32 are mentioned.

	VT_GetTTSInfo_ENG()	▪ Details of VT_DB_BUILD_DATE added.
	<vtml_break>	▪ Level 3 is added.
	<vtml_pause>	▪ Allows final pause.
	<vtml_phoneme>	▪ Appendix D regarding SPR is revised.
	<vtml_sayas>	▪ User instructions are

		complemented.
	Appendix D	▪ Details on x-cmu vowels coupled with Stress are revised.
2007-12 (v3.9.0)	<vtml_phoneme>	▪ alphabet added (x-pinyin)
	Appendix D	▪ x-pinyin description added ▪ pinyin phonetic table added
2008-06 (v3.9.1)	<vtml_phoneme>	▪ IMPORTANT LIMITATION revised.

2008-12 (v3.9.2)	VT_PLAYTTS_ENG()
	VT_TextToFile_ENG()
	VT_TextToBuffer_ENG()
		▪ codepage description added

	VT_GetTTSInfo_ENG() Appendix A	▪ VT_MAX_COMMA_PAUSE added ▪ VT_DEF_COMMA_PAUSE added ▪ VT_MIN_COMMA_PAUSE added
	<vtml_sayas>	▪ sapi:currency output is revised.
	Appendix E	▪ currency, URI, measure, address, symbols expansion are revised.

2009-03 (v3.9.3)	VT_LOADTTS_ENG()
	VT_UNLOADTTS_ENG()
	VT_PLAYTTS_ENG()
	VT_TextToFile_ENG()
	VT_TextToBuffer_ENG()
	VT_SetPitchSpeedVolumePause_ENG()
		▪ Name of TTS voice added (julie)

2009-04 (v3.9.3.2)	<vtml_phoneme>	▪ alphabet added (x-ntsampa)
	Appendix D	▪ x-ntsampa description added ▪ ntsampa phonetic table added ▪ spanish phonetic table added
2009-06 (v3.10.0)	VT_SetCommaPause_ENG()	▪ API added
2009-12 (v3.11.0)	<vtml_phoneme>	▪ ph description revised
2009-12 (v3.11.0)	VT_GetTTSInfo_ENG()	▪ VT_VERIFY_CODE description revised
2010-06 (v3.11.1)	Appendix B	▪ Sample Programs revised
	Appendix D	▪ UK english phonetic table added
2011-06	<vtml_phoneme>	▪ alphabet added (x-tasampa)

(v3.11.3)	<vtml_sayas>	▪ vtml:duration added
	Appendix D	▪ x-tasampa description added ▪ tasampa phonetic table added
2012-06 (v3.11.5)	Appendix D	▪ canadian french phonetic table added

	VT_LOADTTS_ENG()
	VT_UNLOADTTS_ENG()
	VT_PLAYTTS_ENG()
	VT_TextToFile_ENG()
	VT_TextToBuffer_ENG()
	VT_SetPitchSpeedVolumePause_ENG()
	VT_SetCommaPause_ENG()
		▪ Name of TTS voice added (james)

2013-06 (v3.11.7)	Appendix D	▪ spanish phonetic table revised
	Appendix E	▪ Telephone Number revised
	<vtml_sayas>	▪ ssml:telephone revised ▪ sapi:phone revised

	VT_LOADTTS_ENG()
	VT_UNLOADTTS_ENG()
	VT_PLAYTTS_ENG()
	VT_TextToFile_ENG()
	VT_TextToBuffer_ENG()
	VT_SetPitchSpeedVolumePause_ENG()
	VT_SetCommaPause_ENG()
		▪ Name of TTS voice added (ashley)

2015-12
(v3.11.11)
	VT_LOADTTS_ENG()
	VT_UNLOADTTS_ENG()
	VT_PLAYTTS_ENG()
	VT_TextToFile_ENG()
	VT_TextToBuffer_ENG()
	VT_SetPitchSpeedVolumePause_ENG()
	VT_SetCommaPause_ENG()
		▪ Name of TTS voice added (beth)

2016-06 (v3.11.13)	<vtml_phoneme>	▪ Example revised
	Appendix D	▪ phonetic table revised
2017-06 (v3.11.15)	<vtml_phoneme>	▪ worldbet, cmu deleted ▪ lang added ▪ Example revised
	<vtml_sayas>	▪ vtml:number added

	Appendix D	▪ worldbet, cmu description deleted ▪ phonetic table revised ▪ Language Code Table added
2018-06 (v3.12.1)	VT_TextToBuffer_ENG()	▪ Description revised
2019-06 (v3.12.3)	Appendix D：SPR(Symbolic Phonetic Representations)	▪ Symbolic Phonetic Table Deleted

ReadSpeaker speechEngine API

Jun. 2019
Printed in Korea.

TABLE OF CONTENTS

Chapter 1: Overview

ReadSpeaker speechEngine, a Text-to-Speech (TTS) solution by ReadSpeaker is a technology that converts text data into speech. ReadSpeaker speechEngine analyzes the grammatical structure of any text input and generates a very natural-sounding speech output, with natural inflections.

ReadSpeaker speechEngine is a high-quality, high-performance TTS solution, and is capable of handling special text such as numbers and symbols. It is being used in various fields, such as telecommunication, finance, service industries and public organizations. It can create accurate pronunciations and natural voices and supports various platforms. ReadSpeaker speechEngine is a world-class English TTS product, and we are proud of its excellent service stability, fast output generation speed, and maximum number of voices.

This document contains descriptions and examples of ReadSpeaker speechEngine APIs that are necessary for developing software applications using the ReadSpeaker speechEngine.

The Appendix of the manual includes information about VTML(VT Markup Language) Tag Set, which controls inflection (pitch, sound speed, volume, pause), and text preprocessing protocol defined by ReadSpeaker speechEngine.

Chapter 2: Installation

VTAP API consists of header files and libraries, which can be copied to the system for use.

ReadSpeaker speechEngine header files

ReadSpeaker speechEngine Header: vt_eng.h

ReadSpeaker speechEngine Libraries (Win32)

ReadSpeaker speechEngine Library: vt_eng.dll, vt_eng.lib

ReadSpeaker speechEngine Libraries (Unix)

ReadSpeaker speechEngine Library: libvt_eng.a or libvt_eng.so

Compile and Link Procedures
Unix : cc –O2 –o test_prog test_prog.c –lm libvt_eng.a –lpthread –D_REENTRANT
WIN32(MSVC) : Link vt_eng.dll and check and use the '_THREAD_SAFE' under Link Options.

Note
The above files are for Win32 and Unix OS, and may vary for different operating systems.

Chapter 3: Getting Started

ReadSpeaker speechEngine APIs are aimed to facilitate developing applications using the ReadSpeaker speechEngine. The APIs are generally classified into **File APIs** and **Buffering APIs** depending on the method of synthesis. In addition, **Basic APIs** and **Option APIs** are provided for loading, unloading, and setting the synthesis engine. **Information APIs** are used to extract engine-related information. For Win32 systems, **Sound Card APIs** are provided to play synthesized audio through a sound card.

The function name of ReadSpeaker speechEngine API follows the following naming convention:

VT_{function description}_{language of the TTS voice}
(Example: VT_SetPitchSpeedVolumePause_ENG)

Chapter 4: ReadSpeaker speechEngine

This chapter describes ReadSpeaker speechEngine Function References, which can be divided into 5 categories.

- Basic API (loading/unloading the synthesis engine and the User Dictionary)
  VT_LOADTTS_ENG()
  VT_UNLOADTTS_ENG()
  VT_LOAD_UserDict_ENG()
  VT_UNLOAD_UserDict_ENG()

- Sound Card API (Play/Stop of synthesized sound output via sound card)
  VT_PLAYTTS_ENG()
  VT_STOPTTS_ENG()
  VT_PAUSETTS_ENG()
  VT_RESTARTTTS_ENG()

- File API (Synthesize and save to a voice file)
  VT_TextToFile_ENG

- Buffering API (Synthesize to a voice buffer)
  VT_TextToBuffer_ENG()

- Option API (Set/View synthesizer settings)
  VT_SetPitchSpeedVolumePause_ENG()
  VT_SetCommaPause_ENG()

- Information API (Get Engine Information)
  VT_GetTTSInfo_ENG()

VT_LOADTTS_ENG

Loads the synthesizer’s TTS database.

Synopsis

```
#include “vt_eng.h”

short VT_LOADTTS_ENG (
        HWND hWnd
        int nSpeakerID,
        char * db_path,
        char * licensefile);
```

Parameters

hWnd

Always use NULL (for UNIX). Use window handle (for Win32).

nSpeakerID

IDs used to determine the voices when multiple voices are used together.

Following are the IDs used for *nSpeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)
-1	default voice that the engine has defined

db_path

The path where the synthesizer database is located.

NULL	Uses the default value when the engine was installed.

Licensefile

License verification file

NULL	Uses “verification.txt” located on synthesizer database path.

Description

The function loads the synthesizer database and it is used when the program starts.

Return Values

When the database is successfully loaded, it returns 0. The following values are returned when error occurs. (Refer to vt_eng.h)

[1]	Tried to load the same voice from multiple different *db_path*.
[2]	Failed to secure channel memory
[3]	Failed to load DB for the Morpheme Analysis
[4]	Failed to load DB for the Break Index
[5]	Failed to load DB for the Text Pre-Processing
[6]	Failed to load DB for the Acoustic Model
[7]	Failed to load DB for Unit Selection
[8]	Failed to load DB for Prosody Model
[9]	Failed to load DB for Speech Database
[10]	Failed to load DB for Pitch Location Information
[11]	Other errors

Notes

In the case of successful TTS DB load, the return value can be obtained in advance through VT_GetTTSInfo_ENG(). Synthesizer DB and other relative files should be installed as "data-{name of TTS voice}" and "data-common". The names of the speaker-related directory are as follows.

kate(Female)	'data-kate'
paul(Male)	'data-paul'
julie(Female)	'data-julie'
james(Male)	'data-james'
ashley(Female)	'data-ashley'
beth(Female)	'data-beth'

Files

../data-{name of TTS voice}/*

../data-common/*

See Also

VT_UNLOADTTS_ENG()

VT_GetTTSInfo_ENG()

Example

```
    if (VT_LOADTTS_ENG (NULL, -1, NULL, NULL) != VT_LOADTTS_SUCCESS)
        return -1;
```

VT_UNLOADTTS_ENG

Unloads the synthesizer’s DB

Synopsis

#include “vt_eng.h”

void VT_UNLOADTTS_ENG (int nSpeakerID);

Parameters

nSpeakerID

IDs used to determine the voices when multiple voices are used together.

Following are the IDs used for *nSpeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)
-1	default voice that the engine has defined

Description

The function frees the assigned memory by unloading the synthesizer’s DB. It is used at the end of a program.

Return Values

None.

See Also

VT_LOADTTS_ENG()

Example

```
VT_UNLOADTTS_ENG(-1);
```

VT_LOAD_UserDict_ENG

Loads the User Dictionary.

Synopsis

```
#include "vt_eng.h"

short VT_LOAD_UserDict_ENG (
        int dictidx,
        char * filename);
```

Parameters

dictidx

Index used to distinguish dictionaries when multiple user dictionaries are used.
Default user dictionary uses the value of 0 and it can take the values between 1~1023.

filename

The name of user dictionary file.

Description

This function loads the user dictionaries that are separate from and in addition to the default dictionary that is included in the TTS DB.

This has to be used after the completion of loading the synthesizer's DB. *dictidx* is used during synthesis.

Return Values

1 is returned when user dictionary is successfully loaded. The following error codes are returned when errors occur. (refer to vt_eng.h)

[-1]	*dictidx* value is not within the valid range
[-2]	User dictionary file corresponding to *dictidx* is already loaded.
[-3]	Loading failed because there was no a user dictionary file or valid entry
[-4]	Other errors

See Also

VT_UNLOAD_UserDict_ENG()

Example

```
if (VT_LOAD_UserDict_ENG(1, “userdict.csv”) != VT_LOAD_USERDICT_SUCCESS)
    return -1;
```

VT_UNLOAD_UserDict_ENG

Unloads the User Dictionary.

Synopsis

#include “vt_eng.h”

short VT_UNLOAD_UserDict_ENG (int dictidx);

Parameters

dictidx

Index used to distinguish dictionaries when multiple user dictionaries are used.
Default user dictionary uses the value of 0 and it can take the values between 1~1023.

Description

The function unloads the user dictionary.
Any user dictionary that has not been unloaded is automatically unloaded during the process of unloading the synthesizer’s DB.

Return Values

1 is returned when user dictionary is successfully loaded. The following error codes are returned when errors occur. (refer to vt_eng.h)

[-1]	User dictionary file corresponding to *dictidx* is already unloaded.
[-2]	*dictidx* value is not within the valid range
[-3]	Other errors

See Also

VT_LOAD_UserDict_ENG()

Example

```
if (VT_UNLOAD_UserDict_ENG(1) != VT_UNLOAD_USERDICT_SUCCESS)
return -1;
```

VT_PLAYTTS_ENG

Plays synthesized TTS output through a sound card. (Win32 only)

Synopsis

```
#include “vt_eng.h”

short VT_PLAYTTS_ENG (
        HWND hcaller,
        UINT umsg,
        char * text_buff
        int nSpeakerID,
        int pitch,
        int speed,
        int volume,
        int pause,
        int dictidx,
        int texttype);
```

Parameters

hcaller

Windows handle

umsg

User-defined Windows message (WM_USER)

text_buff

Text string to be synthesized - last character must be NULL. (No size limit)

codepage : cp1252

nSpeakerID

IDs used to determine the voices when multiple voices are used together.

Following are the IDs used for *nSpeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)

-1 default voice that the engine has defined

pitch

Defines the pitch of synthesized voice. The default value is set to 100(%). The possible pitch range is 50~200(%) and the lower value indicates the lower pitch. For –1, it uses the default value.

speed

Defines the speed of synthesized voice. The default value is set to 100%. The range is 50~400% and lower value indicates slower speed.

For –1, use default value.

volume

Defines the volume of synthesized voice. The default value is set to 100%. The range is 0~500% and lower value indicates the smaller volume.

For –1, use default value.

pause

Defines the length of pause of synthesized voice. The default value is set to 687(msec). The range is 0~65535(msec)and the lower value indicates shorter pause.

For –1, use default value

dictidx

ID of the user dictionary when multiple user dictionaries are in use. The default user's uses value 0 and the range is between 1~1023.

For –1, use default value.

texttype

This is no longer supported in version 3.6.x and higher. All texts are regarded as plain text.

Description

This is a function that produces output through soundcard by synthesizing input texts. If VT_PLAYTTS_ENG() is called while another TTS output is being played, it overrides the previous output and plays the newly called request.

Once the TTS output is initiated via soundcard, the location of the beginning and the end of the currently-synthesized text is passed to WPARAM and LPARAM, respectively. -1 is passed to LPARAM when the speech playback is completed.

Return Values

1 is returned when it was executed successfully.   The following error codes are returned when errors occur. (refer to vt_eng.h)

[-1]	Failed to secure channel memory
[-2]	The text string is a NULL pointer
[-3]	The length of text string is 0
[-4]	The TTS DB of the voice requested is not loaded
[-5]	Failed to set the sound card
[-6]	Other errors

VT_STOPTTS_ENG

It stops the playback of synthesized voices from a sound card. (Win32 only)

Synopsis

```
#include “vt_eng.h”

void VT_STOPTTS_ENG(void)
```

Parameters

None.

Description

Using VT_PLAYTTS_ENG(), stops the playback of synthesized voice.

Return Values

None.

VT_PAUSETTS_ENG

Pauses the playback of synthesized voices from a sound card. (Win32 only)

Synopsis

```
#include "vt_eng.h"

void VT_PAUSETTS_ENG(void)
```

Parameters

None.

Description

Stops the playback of synthesized voice that is carried out using VT_PLAYTTS_ENG(),

Return Values

None.

VT_RESTARTTTS_ENG

Resumes the playback of the synthesized voice from a sound card.
(Win32 only)

Synopsis

```
#include “vt_eng.h”

void VT_RESTARTTTS_ENG(void)
```

Parameters

None.

Description

Resumes the playback of synthesized voice that was paused using VT_PAUSETTS_ENG().

Return Values

None.

VT_TextToFile_ENG

It saves the synthesized output as a file.

Synopsis

```
#include "vt_eng.h"

short VT_TextToFile_ENG (
        int fmt,
        char * tts_text,
        char * filename,
        int nSpeakerID,
        int pitch,
        int speed,
        int volume,
        int pause,
        int dictidx,
        int texttype);
```

Parameters

fmt

Defines the types of synthesized output formats.

The followings are the types of synthesized output file format that the ReadSpeaker speechEngine supports:

VT_FILE_API_FMT_S16PCM	16bits Linear PCM
VT_FILE_API_FMT_ALAW	8bits A-law PCM
VT_FILE_API_FMT_MULAW	8bits Mu-law PCM
VT_FILE_API_FMT_DADPCM	4bits Dialogic ADPCM
VT_FILE_API_FMT_S16PCM_WAVE	16bits Linear PCM WAVE
VT_FILE_API_FMT_U08PCM_WAVE	8bits Unsigned Linear PCM WAVE
VT_FILE_API_FMT_ALAW_WAVE	8bits A-law PCM WAVE
VT_FILE_API_FMT_MULAW_WAVE	8bits Mu-law PCM WAVE
VT_FILE_API_FMT_MULAW_AU	8bits Mu-law PCM SUN AU

tts_text

Text string to be synthesized - it has to end with NULL.   (no size limit)

codepage : cp1252

filename

File name to save the synthesized voice output under. It is saved under the directory where the program was executed unless assigned separately.

nSpeakerID

Determines the voices when using multiple speakers from the database.

IDs used for *nspeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)

For -1, use the default voice that the engine has defined.

pitch

Defines the pitch of synthesized voice. The default value is set to 100(%). The possible pitch range is 50~200(%) and the lower value indicates the lower pitch. For –1, use default value.

speed

Defines the speed of synthesized voice. The default value is set to 100%. The range is 50~400% and lower value indicates slower speed.

For –1, use default value.

volume

Defines the volume of synthesized voice. The default value is set to 100%. The range is 0~500% and lower value indicates the smaller volume.

For –1, use default value.

pause

Defines the length of pause of synthesized voice. The default value is set to 687(msec). The range is 0~65535(msec)and the lower value indicates shorter pause.

For –1, use default value

dictidx

ID of the user dictionary when multiple user dictionaries are in use. The default user's uses value 0 and the range is between 1~1023.

For –1, use default value.

texttype

This is no longer supported in version 3.6.x and higher. All texts are regarded as plain text.

Description

The synthesize DB must be loaded before using this function.

Return Values

1 is returned when successfully synthesized and the following error codes are returned when errors occur. (refer to vt_eng.h)

[-1]	Used format that is not supported.
[-2]	Failed to secure channel memory
[-3]	Text string is a NULL pointer
[-4]	The length of text string is 0.
[-5]	The TTS DB of the voice requested is not loaded
[-6]	Failed to generate the synthesized voice file
[-7]	Other errors

See Also

VT_LOADTTS_ENG()

VT_GetTTSInfo_ENG()

Example

```
if (VT_TextToFile_ENG (VT_FILE_API_FMT_S16PCM, "Hello?", "test.pcm", -1, -1, -1, -1, -1, -1, -
1) != VT_FILE_API_SUCCESS)
    return -1;
```

VT_TextToBuffer_ENG

It saves the synthesized output as PCM frame buffer.

Synopsis

```
#include “vt_eng.h”

int VT_TextToBuffer_ENG (
        int fmt,
        char * tts_text,
        char * output_buff,
        int * output_len,
        int flag,
        int nThreadID,
        int nSpeakerID,
        int pitch,
        int speed,
        int volume,
        int pause,
        int dictidx,
        int texttype);
```

Parameters

fmt

Defines the type of synthesized output format.

The followings are the synthesized output formats supported by ReadSpeaker speechEngine.

VT_BUFFER_API_FMT_S16PCM	16bits Linear PCM
VT_BUFFER_API_FMT_ALAW	8bits A-law PCM
VT_BUFFER_API_FMT_MULAW	8bits Mu-law PCM
VT_BUFFER_API_FMT_DADPCM	4bits Dialogic ADPCM

tts_text

Text string to be synthesized - string should end with NULL. (no size limit)

codepage : cp1252

output_buff

The pointer to the frame buffer where the synthesized output is being saved.

output_len

The size of synthesized output saved in *output_buff*

flag

Flag that controls the frame processing of the API

nThreadID

ID for identifying threads in case of multiple, concurrent calls of the speech buffer synthesis API function. Values from 0 ~ N-1 should be used, where N is the maximum # of concurrent channels as specified in the license verification file.

nSpeakerID

Determines the voices when using multiple speakers from the database.

IDs used for *nspeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)

For -1, use the default voice that the engine has defined.

pitch

Defines the pitch of synthesized voice. The default value is set to 100(%). The possible pitch range is 50~200(%) and the lower value indicates the lower pitch.

For –1, use default value.

speed

Defines the speed of synthesized voice. The default value is set to 100%. The range is 50~400% and lower value indicates slower speed.

For –1, use default value.

volume

Defines the volume of synthesized voice. The default value is set to 100%. The range is 0~500% and lower value indicates the smaller volume.

For –1, use default value.

pause

Defines the length of pause of synthesized voice. The default value is set to 687(msec). The range is 0~65535(msec)and the lower value indicates shorter pause.

For –1, use default value

dictidx

ID of the user dictionary when multiple user dictionaries are in use. The default user's uses value 0 and the range is between 1~1023.

For –1, use default value.

texttype

This is no longer supported in version 3.6.x and higher. All texts are regarded as plain text.

Description

The Synthesizer DB must be loaded before using this function.

It divides the whole synthesized output into frames with the same size. You can obtain the frames in a streaming fashion through this function. The size of the frame can be found by assigning –1 for *flag*, and the frame size is saved on *output_len* and is returned. It is also returned as the function's return value. Therefore, the frame size supported by API needs to be examined first before taking the first frame.

For example, if the supported frame size is 6kbytes and the total size of synthesized voice output is more than 6kbytes, then the first frame will be 6kbytes and the last frame will be less than 6kbytes. Therefore, the size of output_buff should be more than 6kbytes and output_len should be 6000 or less.

frame 1	frame 2	frame 3	…	frame n
6KB	6KB	6KB	6KB	Below 6KB

flag value should be as follows:

-1 to get the frame size supported,

0 to bring the first frame,

1 to bring the second and further frames, or

2 to abort synthesizing and discard the remaining frames

nThreadID is used for programs that use several voice buffer synthesizing functions at a time. Each function must use a unique number. The number of threads that can be processed at the same time is N (0 to N-1). The verification file specifies the number N.

Return Values

When synthesizing is successful, it returns 1 for the last frame and 0 for the rest of the frames. When error occurs, the following values are returned. (refer to vt_eng.h)

[-1]	Used format that is not supported.
[-2]	Failed to secure channel memory.
[-3]	Text string is a NULL pointer.
[-4]	The length of text string is 0.
[-5]	Frame buffer is NULL pointer.
[-6]	The TTS DB of the voice requested is not loaded
[-7]	The thread ID is already in use.
[-8]	Unknown errors.

See Also

VT_LOADTTS_ENG()

VT_GetTTSInfo_ENG()

Example

```
char *sbuffer;
int slen;
int flag = 0;
int tid = 0;

if (VT_TextToBuffer_ENG (VT_BUFFER_API_FMT_S16PCM, (char *)NULL, (char *)NULL, &slen, -
1, tid, -1, -1, -1, -1, -1, -1, -1) < 0)
    return -1;
sbuffer = (char *) malloc (slen);
while (1) {
    int rc = VT_TextToBuffer_ENG (VT_BUFFER_API_FMT_S16PCM, "Hello?", sbuffer, &slen, flag,
tid , -1, -1, -1, -1, -1, -1, -1);

    if (flag == 0) flag = 1;
    … Play a sbuffer …
    if (rc == 1 || rc < 0) break;
}
free (sbuffer);
```

VT_SetPitchSpeedVolumePause_ENG

It sets the pitch, speed, and volume of synthesized voices and pauses between sentences.

Synopsis

```
#include "vt_eng.h"

void VT_SetPitchSpeedVolumePause_ENG (
        int pitch,
        int speed,
        int volume,
        int pause,
        int nSpeakerID);
```

Parameters

pitch

Defines the pitch of synthesized voice. The default value is set to 100(%). The possible pitch range is 50~200(%) and the lower value indicates the lower pitch. For –1, use default value.

speed

Defines the speed of synthesized voice. The default value is set to 100%. The range is 50~400% and lower value indicates slower speed.

For –1, use default value.

volume

Defines the volume of synthesized voice. The default value is set to 100%. The range is 0~500% and lower value indicates the smaller volume.

For –1, use default value.

pause

Defines the length of pause of synthesized voice. The default value is set to 687(msec). The range is 0~65535(msec)and the lower value indicates shorter pause.

For –1, use default value

nSpeakerID

Determines the voices when using multiple speakers from the database.

IDs used for *nspeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)

For -1, use the default voice that the engine has defined.

Description

It has to be set after loading the synthesizer's database and before synthesizing a voice. The maximum and minimum value of the option can be figured out by using VT_GetTTSInfo_ENG()

Return Values

None.

See Also

VT_LOADTTS_ENG()
VT_GetTTSInfo_ENG()

Example

```
VT_SetPitchSpeedVolumePause_ENG (110, 90, 300, 2000, -1);
```

VT_SetCommaPause_ENG

It sets comma pause.

Synopsis

```
#include “vt_eng.h”

void VT_SetCommaPause_ENG (
        int pause,
        int nSpeakerID);
```

Parameters

pause

Defines the length of comma pause of synthesized voice. The default value is set to 200(msec). The range is 0~65535(msec)and the lower value indicates shorter pause.

For –1, use default value

nSpeakerID

Determines the voices when using multiple speakers from the database.

IDs used for *nspeakerID*

0	kate(Female)
1	paul(Male)
3	julie(Female)
4	james(Male)
5	ashley(Female)
6	beth(Female)

For -1, use the default voice that the engine has defined.

Description

It has to be set after loading the synthesizer’s database and before synthesizing a voice. The maximum and minimum value of the option can be figured out by using VT_GetTTSInfo_ENG()

Return Values

None.

See Also

VT_LOADTTS_ENG()

VT_GetTTSInfo_ENG()

Example

```
VT_SetCommaPause_ENG (2000, -1);
```

VT_GetTTSInfo_ENG

It gets ReadSpeaker speechEngine related information.

Synopsis

```
#include “vt_eng.h”

void VT_GetTTSInfo_ENG (
        int request,
        char * licensefile,
        void * value,
        int valuesize);
```

Parameters

request

Specifies the ReadSpeaker speechEngine information to be extracted

VT_BUILD_DATE	Date when the Engine was built
VT_VERIFY_CODE	Verification Result : Success(0)
VT_MAX_CHANNEL	Number of maximum channels specified on the license verification file
VT_DB_DIRECTORY	Assigned DB directory path when the synthesizer DB was installed
VT_LOAD_SUCCESS_CODE	Return code for successful synthesizer DB loading
VT_MAX_SPEAKER	Maximum number of voices supported
VT_DEF_SPEAKER	Default speaker voice ID
VT_CODEPAGE	code page (Win32) that the engine supports
VT_DB_ACCESS_MODE	Database access mode: File(0), RAM(1)
VT_FIXED_POINT_SUPPORT	Engine’s integer simulation Yes(1)/No(0)
VT_SAMPLING_FREQUENCY	Sampling frequency of voice in Hz (8000, 11025, 16000)
VT_MAX_PITCH_RATE	Maximum pitch value
VT_DEF_PITCH_RATE	Default pitch value
VT_MIN_PITCH_RATE	Minimum pitch value
VT_MAX_SPEED_RATE	Maximum speed value
VT_DEF_SPEED_RATE	Default speed value

VT_MIN_SPEED_RATE	Minimum speed value
VT_MAX_ VOLUME	Maximum volume
VT_DEF_ VOLUME	Default volume
VT_MIN_VOLUME	Minimum volume
VT_MAX_SENT_PAUSE	Maximum sentence pause
VT_DEF_ SENT_PAUSE	Default sentence pause
VT_MIN_ SENT_PAUSE	Minimum sentence pause
VT_DB_BUILD_DATE	DB build date of the embedded engine
VT_MAX_COMMA_PAUSE	Maximum comma pause
VT_DEF_COMMA_PAUSE	Default comma pause
VT_MIN_COMMA_PAUSE	Minimum comma pause

licensefile

License Verification File.

In case of NULL, use "verification.txt" under the database directory

value

The variable pointer that saves the result of *request*

Use (char *) for *request* values of VT_BUILD_DATE, VT_DB_DIRECTORY, VT_DB_BUILD_DATE and use (int *) for other cases.

valuesize

Size of obtained parameter *value* in bytes.

If *value* is (char *), use the value of text string buffer length; if *value* is (int *), set it to 4.

Description

It can be used before loading the TTS DB.

It can be used to check errors when developing TTS-enabled applications by getting various information on what the engine supports.

Return Values

It returns 0 when the information is successfully drawn. It returns the following codes when error occurs. (refer to vt_eng.h)

[1] Unqualified *request* that cannot be used to get any information

[2] Undefined value of *request* is used

[3] *value* is NULL pointer

[4] *valuesize* is too small to be characterize the corresponding attribute/information

[5]	Other errors

See Also

VT_LOADTTS_ENG()

Example

```
int nSampleRate;
if( VT_GetTTSInfo_ENG (VT_SAMPLING_FREQUENCY, NULL, & nSampleRate, sizeof(int) !=
VT_INFO_SUCCESS)
return -1;
```

Appendix A: Header Files

vt_eng.h

```
/*
 * Copyright (c) ReadSpeaker, All rights reserved.
 *
 * ReadSpeaker speechEngine
*/

#ifndef VT_ENG_H
#define VT_ENG_H

#if defined(__cplusplus)
    extern "C" {
#endif

#if !defined(VT_BASIC_DEFINE)
    #if defined(WIN32)
        #if !defined(_DllMode)
            #define _DllMode(_type_)        __declspec( dllimport ) _type_
        #endif
    #else
        #if !defined(_DllMode)
            #define         _DllMode(_type_)                    extern _type_
        #endif
        typedef         int                         HWND;
    #endif
#endif



/*===============================================================*/
/* Text format (used in texttype) */
#if !defined(VT_BASIC_DEFINE)
    #if !defined(VT_TEXT_FMT_PLAIN_TEXT)
        #define    VT_TEXT_FMT_PLAIN_TEXT                                      0
    #endif

    #if !defined(VT_TEXT_FMT_JEITA)
        #define    VT_TEXT_FMT_JEITA                                           4
    #endif

    #if !defined(VT_TEXT_FMT_JEITA_PLUS)
        #define    VT_TEXT_FMT_JEITA_PLUS                                      6
    #endif
#endif



/*===============================================================*/
/* LOAD & UNLOAD */
#if !defined(VT_BASIC_DEFINE)
    /* Return Value */
    #define        VT_LOADTTS_SUCCESS                                          0
    #define        VT_LOADTTS_ERROR_CONFLICT_DBPATH                            1
    #define        VT_LOADTTS_ERROR_TTS_STRUCTURE                              2
    #define        VT_LOADTTS_ERROR_TAGGER                                     3
    #define        VT_LOADTTS_ERROR_BREAK_INDEX                                          4
    #define        VT_LOADTTS_ERROR_TPP_DICT                                             5
    #define        VT_LOADTTS_ERROR_TABLE                                                6
    #define        VT_LOADTTS_ERROR_UNIT_INDEX                                           7
    #define        VT_LOADTTS_ERROR_PROSODY_DB                                           8
    #define        VT_LOADTTS_ERROR_PCM_DB                                               9
    #define        VT_LOADTTS_ERROR_PM_DB                                                10
    #define        VT_LOADTTS_ERROR_UNKNOWN                                              11
#endif

_DllMode(short)  VT_LOADTTS_ENG(HWND hWnd, int nSpeakerID, char *db_path, char *licensefile);
_DllMode(void)   VT_UNLOADTTS_ENG(int nSpeakerID);



/*==========================================================================*/
/* Load/Unload UserDict API */
#if !defined(VT_BASIC_DEFINE)
    /* Return Value */
    #define        VT_LOAD_USERDICT_SUCCESS                                              (1)
    #define        VT_LOAD_USERDICT_ERROR_INVALID_INDEX                                  (-1)
    #define        VT_LOAD_USERDICT_ERROR_INDEX_BUSY                                     (-2)
    #define        VT_LOAD_USERDICT_ERROR_LOAD_FAIL                                      (-3)
    #define        VT_LOAD_USERDICT_ERROR_UNKNOWN                                        (-4)

    #define        VT_UNLOAD_USERDICT_SUCCESS                                            (1)
    #define        VT_UNLOAD_USERDICT_ERROR_NULL_INDEX                                   (-1)
    #define        VT_UNLOAD_USERDICT_ERROR_INVALID_INDEX                                (-2)
    #define        VT_UNLOAD_USERDICT_ERROR_UNKNOWN                                      (-3)
#endif

_DllMode(short) VT_LOAD_UserDict_ENG(int dictidx, char *filename);
_DllMode(short) VT_UNLOAD_UserDict_ENG(int dictidx);



/*==========================================================================*/
/* SOUND CARD API */
#if !defined(VT_BASIC_DEFINE)
    /* Return Value */
    #define        VT_PLAY_API_SUCCESS                                       (1)
    #define        VT_PLAY_API_ERROR_CREATE_THREAD                           (-1)
    #define        VT_PLAY_API_ERROR_NULL_TEXT                               (-2)
    #define        VT_PLAY_API_ERROR_EMPTY_TEXT                              (-3)
    #define        VT_PLAY_API_ERROR_DB_NOT_LOADED                           (-4)
    #define        VT_PLAY_API_ERROR_INITPLAY                                (-5)
    #define        VT_PLAY_API_ERROR_UNKNOWN                                 (-6)
#endif

#if defined(WIN32)
    _DllMode(short) VT_PLAYTTS_ENG(HWND hcaller, UINT umsg, char *text_buff, int nSpeakerID, int
pitch, int speed, int volume, int pause, int dictidx, int texttype);
    _DllMode(void)   VT_STOPTTS_ENG(void);
    _DllMode(void)   VT_RESTARTTTS_ENG(void);
    _DllMode(void)   VT_PAUSETTS_ENG(void);
#endif



/*==========================================================================*/
/* FILE WRITE API */
#if !defined(VT_BASIC_DEFINE)
    /* Return Value */
    #define        VT_FILE_API_SUCCESS                                       (1)
    #define        VT_FILE_API_ERROR_INVALID_FORMAT                          (-1)
    #define        VT_FILE_API_ERROR_CREATE_THREAD                           (-2)
    #define        VT_FILE_API_ERROR_NULL_TEXT                               (-3)
    #define        VT_FILE_API_ERROR_EMPTY_TEXT                                  (-4)
    #define        VT_FILE_API_ERROR_DB_NOT_LOADED                               (-5)
    #define        VT_FILE_API_ERROR_OUT_FILE_OPEN                               (-6)
    #define        VT_FILE_API_ERROR_UNKNOWN                                     (-7)

    /* Audio Format */
    enum {
        VT_FILE_API_FMT_S16PCM      = 0,
        VT_FILE_API_FMT_ALAW        = 1,
        VT_FILE_API_FMT_MULAW       = 2,
        VT_FILE_API_FMT_DADPCM      = 3,
        VT_FILE_API_FMT_S16PCM_WAVE = 4,
        VT_FILE_API_FMT_U08PCM_WAVE = 5,
//  VT_FILE_API_FMT_IMA_WAVE  = 6, /* not supported! */
        VT_FILE_API_FMT_ALAW_WAVE    = 7,
        VT_FILE_API_FMT_MULAW_WAVE  = 8,
        VT_FILE_API_FMT_MULAW_AU = 9,
    };
#endif

_DllMode(short) VT_TextToFile_ENG(int fmt, char *tts_text, char *filename, int nSpeakerID, int pitch,
int speed, int volume, int pause, int dictidx, int texttype);



/*==========================================================================*/
/* BUFFER I/O API */
#if !defined(VT_BASIC_DEFINE)
    /* Return Value */
    #define        VT_BUFFER_API_PROCESSING                                      (0)
    #define        VT_BUFFER_API_DONE                                            (1)
    #define        VT_BUFFER_API_ERROR_INVALID_FORMAT                            (-1)
    #define        VT_BUFFER_API_ERROR_CREATE_THREAD                             (-2)
    #define        VT_BUFFER_API_ERROR_NULL_TEXT                                 (-3)
    #define        VT_BUFFER_API_ERROR_EMPTY_TEXT                                (-4)
    #define        VT_BUFFER_API_ERROR_NULL_BUFFER                               (-5)
    #define        VT_BUFFER_API_ERROR_DB_NOT_LOADED                             (-6)
    #define        VT_BUFFER_API_ERROR_THREAD_BUSY                               (-7)
    #define        VT_BUFFER_API_ERROR_ABNORMAL_CONDITION                        (-8)
    #define        VT_BUFFER_API_ERROR_UNKNOWN                                   (-9)

    /* Audio Format */
    enum {
        VT_BUFFER_API_FMT_S16PCM = VT_FILE_API_FMT_S16PCM,
        VT_BUFFER_API_FMT_ALAW    = VT_FILE_API_FMT_ALAW,
        VT_BUFFER_API_FMT_MULAW   = VT_FILE_API_FMT_MULAW,
        VT_BUFFER_API_FMT_DADPCM = VT_FILE_API_FMT_DADPCM,
    };
#endif

_DllMode(int) VT_TextToBuffer_ENG(int fmt, char *tts_text, char *output_buff, int *output_len, int
flag, int nThreadID, int nSpeakerID, int pitch, int speed, int volume, int pause, int dictidx, int
texttype);



/*==========================================================================*/
/* CONFIGURE API */
_DllMode(void) VT_SetPitchSpeedVolumePause_ENG(int pitch, int speed, int volume, int pause, int
nSpeakerID);
_DllMode(void) VT_SetCommaPause_ENG(int pause, int nSpeakerID);



/*============================================================================
SYNOPSIS
    int VT_GetTTSInfo_ENG(int request, char *licensefile, void *value, int valuesize);
```

```
PARAMETERS
    request
        VT_BUILD_DATE               (char*): library build date
        VT_VERIFY_CODE               (int *): verification result(licensefile is required)
        VT_MAX_CHANNEL               (int *): max no. of possible channels(licensefile is required)
        VT_DB_DIRECTORY             (char*): default root DB fold name
        VT_LOAD_SUCCESS_CODE        (int *): return value, when db loading is success
        VT_MAX_SPEAKER               (int *): max no. of speaker ( >= 0 )
        VT_DEF_SPEAKER               (int *): default speaker id ( >= 0 && < max no. of speaker )
        VT_CODEPAGE                  (int *): supported ansi codepage (WIN32 only)
        VT_DB_ACCESS_MODE            (int *): file or ram i/o ? (file:0, ram:1)
        VT_FIXED_POINT_SUPPORT   (int *): fixed point simulated or not? (float:0, fixed:1)
        VT_SAMPLING_FREQUENCY   (int *): current sampling frequency (8000, 11025, 16000 )
        VT_MAX_PITCH_RATE            (int *): max value of pitch rate (%)
        VT_DEF_PITCH_RATE            (int *): default value of pitch rate (%)
        VT_MIN_PITCH_RATE            (int *): min value of pitch rate (%)
        VT_MAX_SPEED_RATE            (int *): max value of speed rate (%)
        VT_DEF_SPEED_RATE            (int *): default value of speed rate (%)
        VT_MIN_SPEED_RATE            (int *): min value of speed rate (%)
        VT_MAX_VOLUME                (int *): max value of volume (%)
        VT_DEF_VOLUME                (int *): default value of volume (%)
        VT_MIN_VOLUME                (int *): min value of volume (%)
        VT_MAX_SENT_PAUSE            (int *): max value of sentence pause (msec)
        VT_DEF_SENT_PAUSE            (int *): default value of sentence pause (msec)
        VT_MIN_SENT_PAUSE            (int *): min value of sentence pause (msec)
        VT_DB_BUILD_DATE            (char*): embedded db build date (for embedded engine only)
        VT_MAX_COMMA_PAUSE       (int *): max value of comma pause (msec)
        VT_DEF_COMMA_PAUSE       (int *): default value of comma pause (msec)
        VT_MIN_COMMA_PAUSE       (int *): min value of comma pause (msec)

    licensefile
        if NULL, use default licensefile.

    value
        VT_DB_DIRECTORY and VT_BUILD_DATE requests are (char *), and any other request is (int
*)

    valuesize
        maximum length of value in characters

RETURN VALUE
    On success, zero(VT_INFO_SUCCESS) is returned.
    On error, the return value depends on the operation:
        VT_INFO_ERROR_NOT_SUPPORTED_REQUEST     (1)
        VT_INFO_ERROR_INVALID_REQUEST           (2)
        VT_INFO_ERROR_NULL_VALUE                (3)
        VT_INFO_ERROR_SHORT_LENGTH_VALUE        (4)
        VT_INFO_ERROR_UNKNOWN                   (5)
=================================================================*/

#if !defined(VT_BASIC_DEFINE)
    /* Return Value */
    #define   VT_INFO_SUCCESS                             (0)
    #define   VT_INFO_ERROR_NOT_SUPPORTED_REQUEST         (1)
    #define   VT_INFO_ERROR_INVALID_REQUEST               (2)
    #define VT_INFO_ERROR_NULL_VALUE                      (3)
    #define   VT_INFO_ERROR_SHORT_LENGTH_VALUE            (4)
    #define   VT_INFO_ERROR_UNKNOWN                       (5)

    /* Request */
    enum
    {
        VT_BUILD_DATE              =   0,
        VT_VERIFY_CODE           =   1,
        VT_MAX_CHANNEL            =   2,
        VT_DB_DIRECTORY          =   3,
        VT_LOAD_SUCCESS_CODE    =   4,
        VT_MAX_SPEAKER          =   5,
        VT_DEF_SPEAKER          =   6,
        VT_CODEPAGE             =   7,
        VT_DB_ACCESS_MODE       =   8,
        VT_FIXED_POINT_SUPPORT=   9,
        VT_SAMPLING_FREQUENCY = 10,
        VT_MAX_PITCH_RATE       = 11,
        VT_DEF_PITCH_RATE       = 12,
        VT_MIN_PITCH_RATE       = 13,
        VT_MAX_SPEED_RATE       = 14,
        VT_DEF_SPEED_RATE       = 15,
        VT_MIN_SPEED_RATE       = 16,
        VT_MAX_VOLUME           = 17,
        VT_DEF_VOLUME           = 18,
        VT_MIN_VOLUME           = 19,
        VT_MAX_SENT_PAUSE       = 20,
        VT_DEF_SENT_PAUSE       = 21,
        VT_MIN_SENT_PAUSE       = 22,
        VT_DB_BUILD_DATE        = 23,
        VT_MAX_COMMA_PAUSE      = 24,
        VT_DEF_COMMA_PAUSE      = 25,
        VT_MIN_COMMA_PAUSE      = 26,
    };
#endif

_DllMode(int) VT_GetTTSInfo_ENG(int request, char *licensefile, void *value, int valuesize);

#if !defined(VT_BASIC_DEFINE)
    #define VT_BASIC_DEFINE
#endif

#if defined(__cplusplus)
    }
#endif
#endif /* VT_ENG_H */
```

Appendix B: Sample Programs

sample_eng.c

```
/**********************************************/
/* tts sample program                          */
/* made by ReadSpeaker                         */
/**********************************************/
#include <stdio.h>
#include <stdlib.h>

#if defined(WIN32) /* Windows only! */
#include <windows.h>
#endif

#include "../include/vt_eng.h"

int main(int argc, char *argv[])
{
    int infoval;
    int nFramesize;
    char *framebuf;
    int frameflag, rc;
    int nThreadID = 0;
    FILE *fp;

    /****************************************/
    /* TTS Initialize                       */
    /* call VT_LOADTTS_ENG ()               */
    /****************************************/
    if (VT_GetTTSInfo_ENG(VT_LOAD_SUCCESS_CODE, NULL, &infoval, sizeof(int)) !=
VT_INFO_SUCCESS) exit(1);
    if (VT_LOADTTS_ENG((int)NULL, -1, NULL, NULL) != infoval) exit(1);


    /****************************************/
    /* TTS File API                         */
    /* call VT_TextToFile_ENG ()            */
    /****************************************/
    if (VT_TextToFile_ENG (VT_FILE_API_FMT_S16PCM_WAVE , "Hello?", "toFile.wav", -1, -1, -1, -1, -
1, -1, -1) != VT_FILE_API_SUCCESS)
        VT_UNLOADTTS_ENG (-1), exit (1);

    /****************************************/
    /* TTS Buffering API                    */
    /* call VT_TextToBuffer_ENG ()          */
    /****************************************/
    rc = frameflag = -1;
    if (VT_TextToBuffer_ENG (VT_BUFFER_API_FMT_S16PCM , (char *)NULL, (char *)NULL,
&nFramesize, frameflag, nThreadID, -1, -1, -1, -1, -1, -1, -1) < 0)
        VT_UNLOADTTS_ENG (-1), exit (1);
    framebuf = (char *) malloc (nFramesize);
    frameflag++;
    fp=fopen("/tmp/toBuffer.pcm","wb");
    do {
        if ((rc = VT_TextToBuffer_ENG (VT_BUFFER_API_FMT_S16PCM , "Hello?", framebuf,
&nFramesize, frameflag, nThreadID, -1, -1, -1, -1, -1, -1, -1)) < 0)
```

```
            VT_UNLOADTTS_ENG (-1), exit (1);

        if (frameflag == 0) frameflag++;
        fwrite(framebuf,nFramesize,1,fp);
    } while (rc == 0);
    fclose(fp);
    free (framebuf);


    /****************************************/
    /* TTS Exit                             */
    /* call VT_UNLOADTTS_ENG ()             */
    /****************************************/
    VT_UNLOADTTS_ENG (-1);
    exit (0);
}
```

Appendix C: VTML Tagset

<vtml_break>

Description

Sets the Break Indices between words.

Syntax

```
<vtml_break level="0" | "1" | "2" | "3"/>
```

Attributes

Attribute	Description
Level	It sets Break Indices. *Required* (0= read continuously, 1= read with minor break, 2= read with major break, 3=sentence separation)

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

N/A

Example

```
The arrest warrant issued in Florida<vtml_break level="0"/> links the attorney
to a government probe of the Medhyin drug cartel headed up by kingpin Carlos
Later<vtml_break level="2"/> who now serving a life sentence in federal
prison.
```

<vtml_partofsp>

Description

Designates a word class of the word surrounded with tags.

Syntax

```
<vtml_partofsp part="unknown" | "noun" | "verb" | "modifier" | "function" |
"interjection"> text
</vtml_partofsp>
```

Attributes

Attribute	Description
part	Part of speech. *Required*

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

N/A

IMPORTANT LIMITATION

This is applicable only in ReadSpeaker US English speechEngine.

The max length of the text is 512bytes including the NULL character. Anything longer will be truncated.

Example

```
Did you <vtml_partofsp part="verb">record</vtml_partofsp> that
<vtml_partofsp part="noun">record</vtml_partofsp>?
```

<vtml_pause>

Description

Sets a pause to be inserted in the synthesized voice.

Syntax

```
<vtml_pause time="msec"/>
```

Attributes

Attribute	Description
time	Pause length. *Required* The unit is millisecond and it has the value between 0 to 65535. (Anything beyond this value will be set as minimum or maximum value. +, - symbols not usable)

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

N/A

IMPORTANT LIMITATION

Adding pause at the very end of the output is supported by placing this tag at the very end of the text. (From v3.7.2)

Example

```
The arrest warrant issued in Florida<vtml_pause time="1000"/> links the
attorney to a government probe of the Medhyin drug cartel headed
up<vtml_pause time="100"/> by kingpin
Carlos Later who now serving a life sentence in federal prison.
```

<vtml_phoneme>

Description

Sets the phonetic symbols of the texts surrounded with the tags.

Syntax

```
<vtml_phoneme
    ph="string"
    alphabet="ipa" | "x-sampa" | "x-ntsampa" | "x-ntsampa" | "x-sapi"
    lang="string">
  text
</vtml_phoneme>
```

Attributes

Attribute	Description
ph	It represents the pronunciation strings. *Required*
alphabet	It sets the SPR(Symbolic Phonetic Representation) to represent pronunciation strings. *Optional* (if omitted, it will set as "ipa". For detailed information on SPR and TAG usage, please refer to Appendix D.)
lang	It represents 3-letter language code. It can use only if alphabet is "x-ntsampa". *Optional* (if omitted, it will set as current language code. For detailed information on Language Code Table, please refer to Appendix D.)

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

N/A

IMPORTANT LIMITATION

ReadSpeaker US English speechEngine only supports **ipa, x-sampa, x-ntsampa, x-tasampa , x-sapi**.

The max length of the text is 512bytes including the NULL character. Anything longer will be truncated.
For US English synthesizers, the ph value must have less than or equal to **64 phonetic symbols.**

Example

```
<vtml_phoneme alphabet="ipa" ph="116;601;712;109;101;618;116;
111;650;">tomato</vtml_phoneme>
<vtml_phoneme alphabet="x-sampa" ph="t@'meItoU">tomato
</vtml_phoneme>
<vtml_phoneme alphabet="x-ntsampa" ph="t@'meItoU">tomato
</vtml_phoneme>
<vtml_phoneme alphabet="x-tasampa" ph="t@'meItoU">tomato
</vtml_phoneme>
<vtml_phoneme alphabet="x-sapi" ph="h eh - l ow 1">hello
</vtml_phoneme>
```

<vtml_pitch>

Description

Among the prosody information, it sets the pitch of the text surrounded by this tags.

Syntax

```
<vtml_pitch value="pitch">
  child elements
</vtml_pitch>
```

Attributes

Attribute	Description
value	The level of pitch ranging between 50~200%. *Required* (Anything beyond this range will be set as minimum or maximum value. +, - symbols not usable)

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

<vtml_break>, <vtml_partofsp>, <vtml_pause>, <vtml_phoneme>, <vtml_pitch>, <vtml_speed>, <vtml_volume>, <vtml_sayas>, <vtml_sub>

Example

```
<vtml_pitch value="150">The arrest warrant issued in Florida links the attorney
to a government probe of the Medhyin drug cartel headed up by kingpin Carlos
Later who now serving a life sentence in federal prison.</vtml_pitch>
```

<vtml_sayas>

Description

Sets the format of the text.

Syntax

```
<vtml_sayas
    interpret-as="construct_type"
    format="string"
    detail="string">
  text
</vtml_sayas>
```

Attributes

Attribute	Description
interpret-as	Text type. *Required*
format	The format in accordance with the type of text. *Optional* (If omitted, please refer to interpret-as below on how it is handled.)
Detail	Additional information on reading in accordance with the format of the text *Optional*

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

N/A

IMPORTANT LIMITATION

ReadSpeaker speechEngine supports <vtml_sayas> element as stated below
The length of texts is max 512bytes including the NULL character and anything longer will be truncated.

ssml:date

Syntax :

```
        <vtml_sayas interpret-as="ssml:date" format="format">Text</vtml_sayas>
```

Format : mdy, dmy, ymd, md, dm, ym, my, d, m, y

Text : Only numbers, date separators (' / ', ' . ', ' - ') are allowed.

When format is specified, it checks the possible range of each numbers.

Otherwise it regards it in the order of Month, Day, Year.

Example :

Input :

```
<vtml_sayas interpret-as="ssml:date" format="mdy">
        01/02/2007
</vtml_sayas>
```

Output : January 2nd 2007

Input :

```
<vtml_sayas interpret-as="ssml:date" format="dmy">
        01/02/2007
</vtml_sayas>
```

Output : February 1st 2007

Input :

```
<vtml_sayas interpret-as="ssml:date" format="ymd">
        2007/01/02
</vtml_sayas>
```

Output : January 2nd 2007

Input :

```
<vtml_sayas interpret-as="ssml:date" format="md">
        01/02
</vtml_sayas>
```

Output : January 2nd

Input :

```
<vtml_sayas interpret-as="ssml:date" format="dm">
        01/02
</vtml_sayas>
```

Output : February 1st

```
Input   :
<vtml_sayas interpret-as="ssml:date" format="ym">
        2007/01
</vtml_sayas>
Output : January 2007
```

```
Input   :
<vtml_sayas interpret-as="ssml:date" format="my">
        01/2007
</vtml_sayas>
Output : January 2007
```

```
Input   : <vtml_sayas interpret-as="ssml:date" format="d">1</vtml_sayas>
Output : 1st
```

```
Input   : <vtml_sayas interpret-as="ssml:date" format="m">1</vtml_sayas>
Output : January
```

```
Input   : <vtml_sayas interpret-as="ssml:date" format="y">2007</vtml_sayas>
Output : 2007
```

```
Input   : <vtml_sayas interpret-as="ssml:date">01/02/2007</vtml_sayas>
Output : January 2nd 2007
```

ssml:time

Syntax :

```
      <vtml_sayas interpret-as="ssml:time" format="format">Text</vtml_sayas>
```

Format : hms24, hms12

Text :

Only numbers, time separators (' : ', ' . ', empty string), modifier separators (space, empty string), and modifiers ("AM", "A.M.", "am", "a.m.", "A", "a", "PM", "P.M.", "pm", "p.m.", "P", "p") are allowed.

It checks the possible range of each number. When format is not specified, it regards them as hms12. Real numbers are allowed in second.

Restriction : hms12 only permits from 1 to12 for the hour values.

hms24 only permits from 0 to 23 for the hour values.

Example :

Input :

```
<vtml_sayas interpret-as="ssml:time" format="hms12">
        09:21:15
</vtml_sayas>
```

Output : nine twenty one and fifteen seconds

Input :

```
<vtml_sayas interpret-as="ssml:time" format="hms24">
        19:21:30
</vtml_sayas>
```

Output : nineteen twenty one and thirty seconds

Input :

```
<vtml_sayas interpret-as="ssml:time">
        09:21:15
</vtml_sayas>
```

Output : nine twenty one and fifteen seconds

ssml:telephone

Syntax :

```
      <vtml_sayas interpret-as="ssml:telephone" format="format">
        Text
      </vtml_sayas>
```

Format : country code

Text :

Only country code symbol (' + '), numbers, phone number separators (' (', ') ', ' - ', ' . ', ' / ', space, empty string), symbols ('*', '#'), English characters (except Q and Z) are allowed.

Example :

Input :

```
<vtml_sayas interpret-as="ssml:telephone" format="39">
        +39(011)777-7777
</vtml_sayas>
```

Output : 3 9   0 1 1   7 7 7   7 7 7 7

(three nine   zero one one   seven seven seven   seven seven seven seven)

Input   :

```
<vtml_sayas interpret-as="ssml:telephone" format="39">
        +1-800-EXAMPLE
</vtml_sayas>
```

Output : 1   800   3 9 2 6 7 5 3

(one   eight hundred   three nine two six seven five three)

ssml:characters

Syntax :

```
<vtml_sayas interpret-as="ssml:characters" format="format" detail="detail">
        Text
</vtml_sayas>
```

Format : characters

Detail   : The number of characters of the classified group.
(In order to distinguish groups, spaces must be used.)
The sum of characters in all groups should equal to the total number of character in the text.

Text   :

Only English characters, numbers, and 31 1-Byte symbols are allowed.
When Format or Detail is not specified, it will be just spelled out.
The symbols allowed are as follows:

(' ! ', ' # ', ' $ ', ' % ', ' & ', ' ′ ', ' (', ') ', ' * ', ' + ', ' , ', ' - ', ' . ', ' / ', ' : ', ' ; ', ' < ', ' = ', ' > ', ' ? ', ' @ ', ' [', ' \ ', '] ', ' ^ ', ' _ ', ' ` ', ' { ', ' | ', ' } ', ' ~ ')

Example :

Input   :

```
<vtml_sayas interpret-as="ssml:characters" format="characters" detail="3 1 2">
        1a3BZ7
</vtml_sayas>
```

Output : 1a3 B Z7 (one A three   B   Z seven)

Input   :

```
<vtml_sayas interpret-as="ssml:characters" format="characters">
        1a3BZ7
```

</vtml_sayas>
Output : 1a3BZ7 (one A three B Z seven)

Input : <vtml_sayas interpret-as="ssml:characters">1a3BZ7</vtml_sayas>
Output : 1a3BZ7 (one A three B Z seven)

ssml:cardinal

Syntax :

```
<vtml_sayas interpret-as="ssml:cardinal" format="format" detail="detail">
  Text
</vtml_sayas>
```

Format : a symbol
(character which distinguish integers and decimal numbers)

Detail : a symbol
(character which distinguish group the integral part of the number)

Text : Only sign symbols (' + ', ' - '), numbers, Format or detail symbols are allowed.

Restriction : Format and Detail must use different symbols.
The maximum length of Integer Parts is based on (Appendix E:Text preprocessing protocol, Cardinal Numbers condition.)

Example :

Input :

```
<vtml_sayas interpret-as="ssml:cardinal" format=".">
    123.456
</vtml_sayas>
```

Output : 123.456 (one hundred twenty three point four five six)

Input :

```
<vtml_sayas interpret-as="ssml:cardinal" detail=".">
    123.456
</vtml_sayas>
```

Output : 123,456 (one hundred twenty three thousand four hundred fifty six)

Input :

```
<vtml_sayas interpret-as="ssml:cardinal">
    123
```

</vtml_sayas>

Output : 123 (one hundred twenty three)

ssml:ordinal

Syntax : <vtml_sayas interpret-as="ssml:ordinal">Text</vtml_sayas>

Text : Only numbers are allowed.

Restriction : The maximum length is based on (Appendix E:Text preprocessing protocol, Ordinal Numbers condition.)

Example :

Input : <vtml_sayas interpret-as="ssml:ordinal">123</vtml_sayas>

Output : 123th

vtml:duration

Syntax :

```
<vtml_sayas interpret-as="vtml:duration" format="format" detail="detail">
 Text
</vtml_sayas>
```

Format : hms, hm, ms, h, m, s

Text :

Only numbers, time separators (':', '′', '″') are allowed.
It checks the possible range of each number. When format is not specified, it regards them as hms. Real numbers are allowed in second.

Restriction : hms only permits from 0 to 99 for the values.

Example :

Input :

```
<vtml_sayas interpret-as="vtml:duration">
         12:30:50
</vtml_sayas>
```

Output : twelve hours thirty minutes and fifty seconds

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="hms">
         12:30:50
</vtml_sayas>
```

Output : twelve hours thirty minutes and fifty seconds

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="hm">
        12:30
</vtml_sayas>
```

Output : twelve hours thirty minutes

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="ms">
        12:30
</vtml_sayas>
```

Output : twelve minutes and thirty seconds

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="ms">
        12’30”
</vtml_sayas>
```

Output : twelve minutes and thirty seconds

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="h">
        12
</vtml_sayas>
```

Output : twelve hours

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="m">
        30
</vtml_sayas>
```

Output : thirty minutes

Input :

```
<vtml_sayas interpret-as="vtml:duration" format="s">
        50
</vtml_sayas>
```

Output : fifty seconds

vtml:number

Syntax :

```
<vtml_sayas interpret-as="vtml:number" format="format" detail="detail">
 Text
</vtml_sayas>
```

Format : digit(integer Boyomi), decimal(decimal Boyomi)

Detail : 0:oh, 0:zero

If format is "digit" and detail is "0:oh",
- number 0 is set to be read as "oh".

If format is "digit" and detail is "0:zero",
- number 0 is set to be read as "zero".

If format is "decimal" and detail is "0:oh",
- number 0 is set to be read as "oh".

If format is "decimal" and detail is "0:zero",
- number 0 is set to be read as "zero".

Text : Only mark('+', '-') and number are allowed.

Restriction : It only can be supported in **ReadSpeaker US English speechEngine.**
Format and Detail are mandatory field.

Example :

Input :

```
<vtml_sayas interpret-as="vtml:number" format="digit" detail="0:oh">
         0123
</vtml_sayas>
```

Output : oh one two three

```
<vtml_sayas interpret-as="vtml:number" format="digit" detail="0:zero">
         0123
</vtml_sayas>
```

Output : zero one two three

Input :

```
<vtml_sayas interpret-as="vtml:number" format="decimal" detail="0:oh">
         1.0123
</vtml_sayas>
```

Output : one point oh one two three

<vtml_sayas interpret-as="vtml:number" format="decimal" detail="0:zero">
          1.0123
</vtml_sayas>
Output : one point zero one two three

vxml:boolean

Syntax : <vtml_sayas interpret-as="vxml:boolean">Text</vtml_sayas>
Text   : Only true or false value are allowed.
Example :
Input   : <vtml_sayas interpret-as="vxml:boolean">true</vtml_sayas>
Output : true

Input   : <vtml_sayas interpret-as="vxml:boolean">false</vtml_sayas>
Output : false

vxml:date

Syntax : <vtml_sayas interpret-as="vxml:date">Text</vtml_sayas>
Text   : Only yyyymmdd patterned numbers, '?'(fills non-specified dates) are allowed.
Restriction :
yyyy is four digit number representing the year, mm is two digit number representing the month and dd is two digit number representing day.
Example :
Input   : <vtml_sayas interpret-as="vxml:date">20070102</vtml_sayas>
Output : January 2nd 2007

Input   : <vtml_sayas interpret-as="vxml:date">????0102</vtml_sayas>
Output : January 2nd

vxml:digits

Syntax : <vtml_sayas interpret-as="vxml:digits">Text</vtml_sayas>
Text   : Only numbers are allowed.
Example :
Input   : <vtml_sayas interpret-as="vxml:digits">123</vtml_sayas>
Output : 1 2 3 (one two three)

vxml:currency

Syntax : <vtml_sayas interpret-as="vxml:currency">Text</vtml_sayas>

Text : Only UUUmm.nn patterned three digit monetary symbols (UUU), number and decimal point(mm.nn) are allowed.

Example :

Input : <vtml_sayas interpret-as="vxml:currency">USD30.101</vtml_sayas>

Output : 30.101 US dallors

vxml:number

Syntax : <vtml_sayas interpret-as="vxml:number">Text</vtml_sayas>

Text : Only symbols(' + ', ' - '), number, decimal are allowed.

Restriction : The maximum length of Integer Parts is based on (Appendix E:Text preprocessing protocol, Cardinal Numbers condition.)

Example :

Input : <vtml_sayas interpret-as="vxml:number">+123.45</vtml_sayas>

Output : +123.45

vxml:phone

Syntax : <vtml_sayas interpret-as="vxml:phone">Text</vtml_sayas>

Text : Only numbers, 'x'(extension's abbreviation) are allowed.

Example :

Input :

```
<vtml_sayas interpret-as="vxml:phone">
        8005551234x789
</vtml_sayas>
```

Output : 8 0 0 5 5 5 1 2 3 4 extension 7 8 9

(eight zero zero five five five one two three four extension seven eight nine)

vxml:time

Syntax : <vtml_sayas interpret-as="vxml:time">Text</vtml_sayas>

Text :

Only HHMM patterned numbers, 'a'(AM), 'p'(PM), 'h'(24 hour),'?'(ambiguous time of AM/PM) are allowed.

Restriction : HH represents 2 digit hours. MM represents 2 digit minutes and X represents one of 'a', 'p', 'h', '?'. Only when it is 'h', the value from 00 to 23 is permitted as HH value.

Example :

Input : <vtml_sayas interpret-as="vxml:time">0600a</vtml_sayas>

Output : six AM

Input : <vtml_sayas interpret-as="vxml:time">0600p</vtml_sayas>

Output : six PM

Input : <vtml_sayas interpret-as="vxml:time">0600?</vtml_sayas>

Output : six o'clock

Input : <vtml_sayas interpret-as="vxml:time">2310h</vtml_sayas>

Output : twenty three ten

sapi:date

Syntax :

    <vtml_sayas interpret-as="sapi:date" format="format">Text</vtml_sayas>

Format : mdy, dmy, ymd, md, dm, ym, my, y

Text : Only numbers, date separators (' / ', ' . ', ' - ') are allowed.

If format is specified, it checks the possible range of numbers.

Otherwise it regards it in the order of month, day, year.

Example :

Input :

```
<vtml_sayas interpret-as="sapi:date" format="mdy">
        01/02/2007
</vtml_sayas>
```

Output : January 2nd 2007

Input :

```
<vtml_sayas interpret-as="sapi:date" format="dmy">
        01/02/2007
</vtml_sayas>
```

Output : February 1st 2007

Input :

```
<vtml_sayas interpret-as="sapi:date" format="ymd">
        2007/01/02
```

```
</vtml_sayas>
```
Output : January 2nd 2007

Input :
```
<vtml_sayas interpret-as="sapi:date" format="md">
          01/02
</vtml_sayas>
```
Output : January 2nd

Input :
```
<vtml_sayas interpret-as="sapi:date" format="dm">
          01/02
</vtml_sayas>
```
Output : February 1st

Input :
```
<vtml_sayas interpret-as="sapi:date" format="ym">
          2007/01
</vtml_sayas>
```
Output : January 2007

Input :
```
<vtml_sayas interpret-as="sapi:date" format="my">
          01/2007
</vtml_sayas>
```
Output : January 2007

Input : `<vtml_sayas interpret-as="sapi:date" format="y">2007</vtml_sayas>`
Output : 2007

Input : `<vtml_sayas interpret-as="sapi:date">01/02/2007</vtml_sayas>`
Output : January 2nd 2007

sapi:time
Syntax : `<vtml_sayas interpret-as="sapi:time">Text</vtml_sayas>`
Text : Only numbers, time separators (' : ', ' ′ ', ' ″ ') are allowed.

It checks the possible range of numbers.

Example :

Input : <vtml_sayas interpret-as="sapi:time">09:21:15</vtml_sayas>

Output : nine twenty one and fifteen seconds

Input : <vtml_sayas interpret-as="sapi:time">1'21"</vtml_sayas>

Output : one minute and twenty one seconds

sapi:number

Syntax :

```
        <vtml_sayas interpret-as="sapi:number" format="format">
          Text
        </vtml_sayas>
```

Format : cardinal, digit, fraction, decimal

Text : Only numbers, decimal point, fraction symbol (' / ') are allowed.
Even if Format is not specified, it will be regarded as cardinal.

Restriction : The maximum length of Integer Parts is based on (Appendix E:Text preprocessing protocol, Cardinal Numbers condition.)

Example :

Input :

```
<vtml_sayas interpret-as="sapi:number" format="cardinal">
          3432
</vtml_sayas>
```

Outpt : 3432 (three thousand four hundred thirty two)

Input :

```
<vtml_sayas interpret-as="sapi:number" format="fraction">
          3/15
</vtml_sayas>
```

Output : 3/15 (three fifteenth)

Input :

```
<vtml_sayas interpret-as="sapi:number" format="digit">
          123
</vtml_sayas>
```

Output : 1 2 3 (one two three)

```
Input   :
<vtml_sayas interpret-as="sapi:number" format="decimal">
         123.456
</vtml_sayas>
Output : 123.456 ( one hundred twenty three point four five six )
```

```
Input   :
<vtml_sayas interpret-as="sapi:number">
         3432
</vtml_sayas>
Outpt   : 3432 ( three thousand four hundred thirty two )
```

sapi:phone

Syntax : <vtml_sayas interpret-as="sapi:phone">Text</vtml_sayas>

Text : Only country code symbol (' + '), numbers,
phone number separator (' - ') are allowed.

Example :

```
Input   :
<vtml_sayas interpret-as="sapi:phone">
         +82-02-3016-8541
</vtml_sayas>
Output : 8 2   0 2   3 0 1 6   8 5 4 1
         ( eight two   zero two   three zero one six   eight five four one )
```

sapi:currency

Syntax : <vtml_sayas interpret-as="sapi:currency">Text</vtml_sayas>

Text : Only currency symbols ('$', '£',…), numbers, decimal point are allowed.

Example :

```
Input   : <vtml_sayas interpret-as="sapi:currency">$34.90</vtml_sayas>
Output : 34.90 dollars ( thirty four dollars ninety cents )
```

sapi:web

Syntax :

```
       <vtml_sayas interpret-as="sapi:web" format="format">Text</vtml_sayas>
```

Format : url

Text   : Only English characters, numbers, symbols (' : ', ' / ', ' . ', ' _ ', ' - ') are allowed.
Even if Format is not specified, it will be regarded as url.

Example :

Input   :

```
<vtml_sayas interpret-as="sapi:web" format="url">
        www.Microsoft.com
</vtml_sayas>
```

Output : W W W dot Microsoft dot com

Input   :

```
<vtml_sayas interpret-as="sapi:web">
        NBA.com
</vtml_sayas>
```

Output : N B A dot com

sapi:email

Syntax : <vtml_sayas interpret-as="sapi:email">Text</vtml_sayas>

Text   : Only English characters, numbers, symbols ('@', ' _ ', ' . ') are allowed.

Example :

Input   :

```
<vtml_sayas interpret-as="sapi:email">
        someone@microsoft.com
</vtml_sayas>
```

Output : someone at Microsoft dot com

sapi:address

Syntax :

```
      <vtml_sayas interpret-as="sapi:address" format="format">
        Text
      </vtml_sayas>
```

Format : postal

Text   : Only English characters, numbers, symbols (' # ', ' , ', ' . ', ' - ') are allowed.

Restriction : Only USA formatted addresses are allowed.
Only Canada and USA formatted postal addresses are allowed.

Example :

Input :

```
<vtml_sayas interpret-as="sapi:address" format="postal">
          A2C 4X5
</vtml_sayas>
```

Output : A 2 C   4 X 5 (A two C   four X five)

Input :

```
<vtml_sayas interpret-as="sapi:address">
          One Microsoft Way, Redmond, WA, 98052
</vtml_sayas>
```

Output : One Microsoft Way   Redmond   Washington nine eight oh five two

Example

```
<vtml_sayas interpret-as="ssml:characters" format="characters">VoiceXML
</vtml_sayas>
You still owe me <vtml_sayas interpret-as="vxml:currency">USD30.10
</vtml_sayas>
Today's date is <vtml_sayas interpret-as="ssml:date" format="mdy">01/02/2006
</vtml_sayas>
Please push the <vtml_sayas interpret-as="vxml:boolean">true</vtml_sayas>
button.
I will get there at <vtml_sayas interpret-as="ssml:time" format="hms24">
07:30:30.0PM</vtml_sayas>
```

<vtml_speed>

Description

Among the prosody information, it sets the speed of text surrounded with this tags.

Syntax

```
<vtml_speed value="speed">
  child elements
</vtml_speed>
```

Attributes

Attribute	Description
speed	It sets the utterance speed with 50~400(%) value. *Required* (Anything beyond this range will be set minimum or maximum value. +, - symbols not usable)

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

<vtml_break>, <vtml_partofsp>, <vtml_pause>, <vtml_phoneme>, <vtml_pitch>, <vtml_speed>, <vtml_volume>, <vtml_sayas>, <vtml_sub>

Example

```
<vtml_speed value="150">The arrest warrant issued in Florida links the attorney
to a government probe of the Medhyin drug cartel headed up by kingpin Carlos
Later who now serving a life sentence in federal prison.</vtml_speed>
```

<vtml_sub>

Description

Reads text by replacing text surrounded with tags with the alias value.

Syntax

```
<vtml_sub
     alias="string">
  text
</vtml_sub>
```

Attributes

Attribute	Description
alias	It replaces the value surrounded with tags. *Required*

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

N/A

IMPORTANT LIMITATION

The length of the texts is max 512bytes including the NULL character. Anything longer will be truncated.
The length of the alias is max 512bytes including the NULL character. Anything longer will be considered as tag error.

Example

```
<vtml_sub alias="World Wide Web Consortium">W3C</vtml_sub>
```

<vtml_volume>

Description

Sets the volume of the text's prosody information surrounded with tags.

Syntax

```
<vtml_volume value="volume">
  child elements
</vtml_volume>
```

Attributes

Attribute	Description
volume	Set the volume within 0 to 500(%) value. *Required* (Anything beyond this range will be set as minimum or maximum value. +, - symbols not usable)

Parents

<vtml_pitch>, <vtml_speed>, <vtml_volume>

Children

<vtml_break>, <vtml_partofsp>, <vtml_pause>, <vtml_phoneme>, <vtml_pitch>, <vtml_speed>, <vtml_volume>, <vtml_sayas>, <vtml_sub>

Example

```
<vtml_volume value="150">The arrest warrant issued in Florida links the attorney
to a government probe of the Medhyin drug cartel headed up by kingpin Carlos
Later who now serving a life sentence in federal prison.</vtml_volume>
```

Appendix D: SPR(Symbolic Phonetic Representations)

Symbolic Phonetic Representation is only supported in **ReadSpeaker US English speechEngine** and the method of expressing the phonetic representation varies with each <vtml_phoneme>tag's optional alphabet.

US English synthesizer supports only the below mentioned five types such as ipa, x-sampa, x-ntsampa, x-tasampa, x-sapi. Each phonetic representation which can be used in each alphabet is a concise English version and defined in Symbolic Phonetic Table. For detailed information on Symbolic Phonetic Table, please refer to **ReadSpeaker Phonetic Symbol Tables Reference Guide** v1.0.

ipa : It is a method of expressing International Phonetic Alphabet-defined phonetic representation as Unicode and the alphabet value **"ipa" must be small letters**. The Unicode must be put in decimal numbers and semi-colon must be put at the end of it.
ex)

```
<vtml_phonemealphabet="ipa" ph="116;601;712;109;101;618;116;
111;650;">tomato</vtml_phoneme>
```

x-SAMPA : It is a method of expressing Speech Assessment Methods Phonetic Alphabet-defined ASCII phonetic representation and the prefix **"x-" must be small letter.**
ex)

```
<vtml_phoneme alphabet="x-sampa" ph="t@'meItoU">tomato
</vtml_phoneme>
```

x-NTSAMPA : It is a method of expressing NAVTEQ-defined ASCII phonetic representation and the prefix **"x-" must be small letter.**
ex)

```
<vtml_phoneme alphabet="x-ntsampa" ph="t@'meItoU">tomato
</vtml_phoneme>
```

x-TASAMPA : It is a method of expressing TELEATLAS-defined ASCII phonetic representation and the prefix **"x-" must be small letter.**

ex)

```
<vtml_phoneme alphabet="x-tasampa" ph="t@'meItoU">tomato
</vtml_phoneme>
```

x-SAPI : It's a method of expressing phonetic representation defined by SAPI Phoneme Representation and the prefix **"x-" must be small letter.**

ex)

```
<vtml_phoneme alphabet="x-sapi" ph="h eh - l ow 1">hello
</vtml_phoneme>
```

Language Code Table

Language Name	Language Code
AUSTRALIA ENGLISH	AUE
BRAZIL PORTUGUESE	BPT
CANADA FRENCH	CFR
FRANCE FRENCH	FRE
GERMANY GERMAN	GER
INDONESIA INDONESIAN	IND
ITALY ITALIAN	ITA
MEXICO SPANISH	SPA
NETHERLANDS DUTCH	DUT
NORWAY NORWEGIAN	NOR
PORTUGAL PORTUGUESE	POR
ROMANIA ROMANIAN	RON
RUSSIA RUSSIAN	RUS
SLOVAKIA SLOVAK	SLK
SPAIN SPANISH	ESP
SWEDEN SWEDISH	SWE
THAILAND THAI	THA
UK ENGLISH	BRE
US ENGLISH	ENG

Appendix E: Text preprocessing protocol

1. Numbers

1. 1 Cardinal Numbers

Ordinary method of reading number. **However, a number with 16 digits or more and number beginning with 0 will be read by each digit.**

Example	Expansion
123	one hundred twenty three
12,345	twelve thousand three hundred forty five
12345	twelve thousand three hundred forty five

1.2 Ordinal Numbers

It follows ordinal number reading. **However, a number with 16 digits or more will be read by each digit.**

Example	Expansion
123rd	one hundred twenty third
12,345th	twelve thousand three hundred forty fifth

1.3 Decimal Fractions

It separates then reads numbers of left and right of the decimal point.

Example	Expansion
1.23	one point two three
.123	point one two three

1.4 Fractions

Example	Expansion
⅔	two thirds
3 ⅔	three and two thirds

1.5 Digit Mode

It reads digit by digit regardless of cipher.

Example	Expansion
+82-2-1234-4567	plus eight two, two, one two three four, four five six seven
157-26-5734	one five seven dash two six dash five seven three four

1.6 Numbers used in product names

The way of reading varies by cipher.

Example	Expansion
LS1234	L S twelve thirty four
F-306	F three oh six

2. Date

Date is read in the order of month, day, year.

Example	Expansion
98/12/01	December first ninety eight
Sept. 11, 2004	September eleventh two thousand four
Mon, the 1st of May.	Monday the first of May.
4-'03	April two thousand three

3. Time

Time is read in the order of hour, minute, second, and time related words.

Example	Expansion
01:12:34	one twelve and thirty four seconds
01:12:34 EST,	one twelve and thirty four seconds Eastern Standard Time

01:12:34 am., e.s.t.,	one twelve and thirty four seconds A M, Eastern Standard Time

4. Currency

Currency units are read according to singular or plural form.

Example	Expansion
$10.09	ten dollars and nine cents
$10.5	ten point five dollars
$10.12US	ten U S dollars and twelve cents
$US10.50	ten U S dollars and fifty cents

5. E-Mail & URI

5.1 E-Mail

Example	Expansion
myid@my.com	myid at M Y dot com
mduerst@ifi.unizh.ch	mduerst at ifi dot U N I Z H dot C H

5.2 URI

Example	Expansion
http://www.readspeaker.com	H T T P colon slash slash W W W dot readspeaker dot com
ftp://ds.internic.net/rfc/	F T P colon slash slash D S dot internic dot net slash R F C slash

6. Telephone Number

Telephone numbers are read in the order of nation number, regional number, telephone exchange number, subscriber number and extension number digit by digit.

Example	Expansion
1 800 260 2650	one, eight hundred, two six zero, two six five zero
02.3016.8541	zero two, three zero one six, eight five four one
02.3016.8541 Ext. 15	zero two, three zero one six, eight five four one, extension one five
Call me at 337-4291	Call me at three three seven, four two nine one

7. Social Security Number

It is read digit by digit.

Example	Expansion
157-26-5734	one five seven dash two six dash five seven three four
690823-2274321	six nine oh eight two three dash two two seven four three two one
12-1234567	one two dash one two three four five six seven

8. Mathematics Operator

The basic operators used in arithmetic are read differently according to their usages.

Example	Expansion
1 + 5 = 6	one plus five equals six
2 * 6 = 12	two times six equals twelve
1 - 90℃.	one dash ninety degrees celsius
1 - 3/5	one minus three fifths
95 mile/h	ninety five mile per hour
3 3/4	three and three quarters

9. Measure

It expands abbreviations used as measure through analyzing neighboring context.

Example	Expansion
$\boldsymbol{t}^4$	ton to the fourth
$\boldsymbol{l}/m^2$	liter per meter squared

1kg + 35kg	one kilogram plus thirty five kilograms
18° ***f***	eighteen degrees Fahrenheit
The length is 10***m***.	The length is ten meters.

10. Address

Addresses are read in the order of Primary Street, Secondary Street, Post-Office Box, City, State & Zip-code.
It follows the U.S. and Canadian address reading method.

Example	Expansion
2381 Dutch Fork Rd. Chapin, SC 29036	twenty three eighty one Dutch Fork Road, Chapin, South Carolina, two nine oh three six
2005 Pan Am Cir Ste. 800 Tampa, FL 33607	two thousand five Pan Am Circle Suite eight hundred, Tampa, Florida, three three six oh seven
Voice Response, Inc. One Corporate E. 1910 E. Kimberly Pl. Montreal , QC L7C 4P8	Voice Response, Incorporated One Corporate E. nineteen ten East Kimberly Place Montreal , Quebec L seven C four P eight

11. Title & Person Name

Abbreviated title/name expands through analyzing neighboring context.

Example	Expansion
Capt. ***Wm***. ***O***. Barnett	Captain William O Barnett
Jill ***St***. John lives on	Jill Saint John lives on
2nd Lt. ***Jas***. Keil	second Lieutenant James Keil

12. Symbols

Specific symbols' reading method is set by the neighboring context. 2 byte symbols have no ambiguities therefore they are read according to the symbols.

Example	Expansion
10 ~ 20	ten to twenty

3.5 > 2.5	three point five greater than two point five
X^1	X to the first
ⅰ	one
⒜	A

13. Abbreviations

Widely used abbreviations or abbreviations without ambiguity on expansion always expand regardless of the context.

Example	Expansion	Example	Expansion
bldg	Building	c.f	compare
corp	corporation	vs	versus